Warszawa, 2.06.2017 r.

WDWR

Projekt

nr 17602

Stawik Małgorzata
258883

Prowadzący
Dr inż. Adam Krzemienowski

Spis treści

1 Wstep

W danym projekcie wykorzystane zostało środowisko *R-statistics* w celu obliczenia wartości oczekiwanych zawężonych rozkładów brzegowych wg wzoru (2.2) oraz aby wygenerować scenariusze do realizacji zadania 2 na podstawie wzoru (3.3).

Natomiast w środowisku *NetBeans IDE* w języku java przy zastosowaniu biblioteki *Cplex* rozwiązane zostało zadanie 1 oraz 2.

2 Zadanie 1

Zaproponować jednokryterialny model wyboru w warunkach ryzyka z wartością oczekiwaną jako miarą kosztu. Wyznaczyć rozwiązanie optymalne.

2.1 Dane

Na rysunku 2.1 przedstawiona została wydajność maszyn (szt./godz.) przy produkcji danego podzespołu.

Maszyna	Wydajność maszyny (szt./godz.) przy produkcji podzespołu				
	A	B	C	D	E
M1	0,85	1,30	0,65	1,50	0,40
M2	0,65	0,80	0,55	1,50	0,70
M3	1,20	0,95	0,35	1,70	0,40

Rysunek 2.1: Wydajność maszyny dla danego podzespołu.

Koszt dzierżawy każdej z maszyn jest różny. Cenę 1 godziny wyrażonej w złotówkach określają składowe wektora losowego $\mathbf{R} = (R_1, R_2, R_3)^T$, który odpowiada kolejno maszynie M1, M2 oraz M3.

Wektor losowy $\mathbf{R}$ opisuje 3-wymiarowy rozkład t-Studenta z 4 stopniami swobody, którego wartości składowych zostały zawężone do przedziału [20,50]. Wektor wartości oczekiwanych μ oraz macierz kowariancji Σ zostały przedstawione we wzorze (2.1).

$$\mu = \begin{pmatrix} 45 \\ 35 \\ 40 \end{pmatrix}, \qquad \Sigma = \begin{pmatrix} 1 & -2 & -1 \\ -2 & 36 & -8 \\ -1 & -8 & 9 \end{pmatrix} \tag{2.1}$$

Wartości oczekiwane zawężonych rozkładów brzegowych zostały wyznaczone na podstawie wzoru (2.2), ponieważ rozkład t-Studenta jest ciągły, więc wartość oczekiwana na przedziale domkniętnym jest taka sama jak na ciągłym.

$$\mathbb{E}(R) = \mu + \sigma * \frac{\Gamma((v-1)/2)((v+a^2)^{-(v-1)/2} - (v+b^2)^{-(v-1)/2})v^{v/2}}{2(F_v(b) - F_v(b))\Gamma(v/2)\Gamma(1/2)} \quad \text{dla } v > 1 \quad (2.2)$$

Gdzie:
- $a = (\alpha - \mu)/\sigma$,
- α - lewy kraniec przedziału,
- $b = (\beta - \mu)/\sigma$,
- β - prawy kraniec przedziału,
- v - liczba stopni swobody.

Podstawiając zmienne do wzoru (1.2) otrzymano:

$\mathbb{E}(R_1) \approx 44{,}97$,
$\mathbb{E}(R_2) \approx 35.23$,
$\mathbb{E}(R_3) \approx 39.83$.

2.2 Model

Zmienne:

- *bool*[k] - zmienna binarna,
- *cost*[k] - koszt dzierżawy za 1 godzinę pracy k-tej maszyny,
- *costAvg*[k] - średni koszt k-tej maszyny,
- *costExtra*[k] - zmienna wynosząca 0 lub 20% kosztu dzierżawy za 1 godzinę pracy k-tej maszyny,
- *costScenario*[t][k] - koszt wygenerowanego scenariusza t dla k-tej maszyny.
- *hours*[k] - liczba godzin pracy k-tej maszyny,
- *items*[k][i] - ilość sztuk k-tej maszyny, i-tego podzespołu (kolejno A, B, C, D oraz E),
- *intItems*[k][i] - zmienna pomocnicza wymuszająca wartości całkowite tablicy *items*[k][i],
- *efficiency*[k][i] - wydajność (szt./godz.) k-tej maszyny, i-tego podzespołu na podstawie rysunku 2.1.

2.2.1 Założenia zadania

1. Maszyny powinny produkować całe sztuki, za pomocą zmiennej całkowitej *intItems*[k][i] wymuszamy by liczba wyprodukowanych sztuk była liczbą całkowitą.

$$\bigwedge_k \bigwedge_i \quad intItems[k][i] \leq items[k][i] \quad (2.3)$$

$$\bigwedge_k \bigwedge_i \quad intItems[k][i] \geq items[k][i] - 1 \tag{2.4}$$

$$\bigwedge_k \bigwedge_i \quad intItems[k][i] = items[k][i] \tag{2.5}$$

Na podstawie powyższych równań wymuszamy na zmiennej $items[k][i]$ odpowiadającej za liczbę sztuk pewnego podzespołu by przyjmowała górne lub dolne zaokrąglenie do najbliższej wartości całkowitej.

2. Do produkcji pięciu podzespołów (A, B, C, D i E) przedsiębiorstwo musi wydzierżawić trzy maszyny, czyli każda maszyna musi wyprodukować co najmniej jedną sztukę z któregokolwiek podzespołu.
Wymóg ten został przedstawiony w nierówności (2.6) dla k-tej maszyny.

$$\bigwedge_k \quad \sum_{i=1}^{5} items[k][i] \geq 1 \tag{2.6}$$

3. Każdą z maszyn można wydzierżawić na co najwyżej 180 godzin w miesiącu.
Liczba godzin pracy każdej z maszyn została przedstawiona wzorem (2.7) jako iloczyn wyprodukowanych sztuk danego podzespołu oraz odwrotności wydajności (czas potrzebny na wyprodukowanie jednej sztuki wyrażony w godz.).
Dane ograniczenie wyrażone zostało nierównością (2.8), gdzie suma czasu potrzebnego do wyprodukowania odpowiednich sztuk danego podzespołu nie może przekroczyć 180 godzin dla k-tej maszyny.

$$\bigwedge_k \quad hours[k] = \sum_{i=1}^{5} \frac{1}{efficiency[k][i]} * items[k][i] \tag{2.7}$$

$$\bigwedge_k \quad hours[k] \leq 180 \tag{2.8}$$

4. Należy rozdzielić miesięczną produkcję podzespołów pomiędzy maszyny tak, aby wyprodukować co najmniej po 60 sztuk podzespołów A, B, C oraz 120 sztuk podzespołów D i E. Pierwszy wymóg przedstawia ograniczenie (2.9) dla k-tej maszyny, gdzie i-ta zmienna odpowiada kolejno podzespołowi A, B oraz C.

$$\sum_{k=1}^{3} items[k][i] \geq 60 \quad \text{dla } i = 1, 2 \text{ oraz } 3. \tag{2.9}$$

Z kolei wymóg na 120 sztuk podzespołu D oraz E przedstawia nierówność (2.10).

$$\sum_{k=1}^{3} \sum_{i=4}^{5} items[k][i] \geq 120 \tag{2.10}$$

5. Przy dzierżawie dowolnej z maszyn na okres do 100 godzin, koszt 1 godziny pracy maszyny maleje o 20%.
W tym założeniu wykorzystana została zmienna binarna $bool[k]$, przyjmująca wartości {0,1} odpowiednio dla k-tej maszyny.
Zgodnie z wyrażeniem (2.11) suma zmiennej $bool[k]$ nie przekroczy liczby maszyn.

$$\sum_{k=1}^{3} bool[k][i] \leq 3 \tag{2.11}$$

$$\bigwedge_{k} \quad hours[k] \leq 100 + \mathrm{M} * bool[k] \tag{2.12}$$

$$\bigwedge_{k} \quad hours[k] \geq 100 * bool[k] \tag{2.13}$$

Gdzie M jest odpowiednio dużą liczbą.
W nierówności (2.12) wartość M wynosi 180 - 100 = 80. Liczba 180 odpowiada maksymalnej liczbie godzin pracy k-tej maszyny.

W przypadku gdy liczba godzin pracy k tej maszyny jest większa od 100 zmienna $bool[k]$ wynosi 1, w przeciwnym przypadku jest równa 0.

W związku z niemożnością mnożenia równań przez zmienne binarne wykorzystana została zmienna pomocnicza ograniczona nierównościami wg (2.14).

$$\bigwedge_{k} \quad L \leq 0,2 * cost[k] * godz[k] \leq U \tag{2.14}$$

$$\bigwedge_{k} \quad costExtra[k] \leq U * bool[k] \tag{2.15}$$

$$\bigwedge_{k} \quad costExtra[k] \geq L * bool[k] \tag{2.16}$$

$$\bigwedge_{k} \quad costExtra[k] \leq 0,2 * cost[k] * godz[k] \tag{2.17}$$

$$\bigwedge_{k} \quad costExtra[k] + U * (1 - bool[k]) \geq 0,2 * cost[k] * godz[k] \tag{2.18}$$

Na potrzeby zadania:
L = 0,2*$cost[k]$,
U = 0,2*$cost[k]$*max($godz[k]$)
Maksymalna ilość godzin osiągalna w zadaniu wynosi 180, dlatego U = 36*$cost[k]$.

2.2.2 Rozwiązanie

Rozwiązaniem zadania jest minimalizacja kosztów przy spełnieniu założeń zadania wg wzoru (2.19)

$$min \quad \sum_{k=1}^{3} 0,8 * cost[k] * godz[k] + costExtra[k] \tag{2.19}$$

Zgodnie z założeniem nr 5 w poprzednim podrozdziale zmienna *bool[k]* przyjmuje wartość 1, gdy czas pracy maszyny przekracza 100 godzin, a co za tym idzie klient płaci 100% kosztów.
W przeciwnym przypadku koszt dzierżawy maszyny za godzinę wynosi 80% ceny początkowej, przy minimalizacji kosztu - efekt pożądany.

Otrzymano rozwiązanie kosztu równe: 8653,48.

Liczba sztuk każdego wyprodukowanego podzespołu:

Maszyna	Liczba wyprodukowanych sztuk				
	A	B	C	D	E
M1	0,00	60,00	18,00	0,00	0,00
M2	0,00	0,00	42,00	35,00	0,00
M3	60,00	0,00	0,00	85,00	0,00

Rysunek 2.2: Liczba sztuk wyprodukowanego podzespołu podzespołu.

Wektor godzin pracy k-tej maszyny:

$$hours[k] = [73,85; 99,70; 100,0] \tag{2.20}$$

Z równania (2.20) wynika wektor $bool[k]$:

$$bool[k] = [0; 0; 0] \tag{2.21}$$

Czyli ostatecznie $costExtra[k]$:

$$costExtra[k] = [0,0; 0,0; 0,0] \tag{2.22}$$

3 Zadanie 2

Jako rozszerzenie zadania 1 zaproponować dwukryterialny model kosztu i ryzyka ze średnią jaką miarą kosztu i średnią różnicą Giniego jako miarą ryzyka.
Dla decyzji $x \in Q$ średnia różnica Giniego przedstawiona została wzorem (3.1).

$$\Gamma(\mathbf{x}) = \frac{1}{2} \sum_{t'=1}^{T} \sum_{t''=1}^{T} |r_{t'}(\mathbf{x}) - r_{t''}(\mathbf{x})| p_{t'} p_{t''} \tag{3.1}$$

Gdzie:
- $r_t(\mathbf{x})$ - realizacja scenariusza t,
- p_t - prawdopodobieństwo scenariusza t.

Z kolei średnią kosztu pracy k-tej maszyny przedstawia poniższy wzór:

$$\bigwedge_k \quad costAvg[k] = \frac{1}{T} \sum_{t=1}^{T} costScenario[t][k] \tag{3.2}$$

Scenariusze potrzebne do rozwiązania danego zadania zostały wygenerowane przy użyciu poniżej funkcji *rtmvt* dostępnej w pakiecie *tmvtnorm* w programie *R-statistics*.

$$rtmvt(n, \mu, \Sigma, df, lower, upper) \tag{3.3}$$

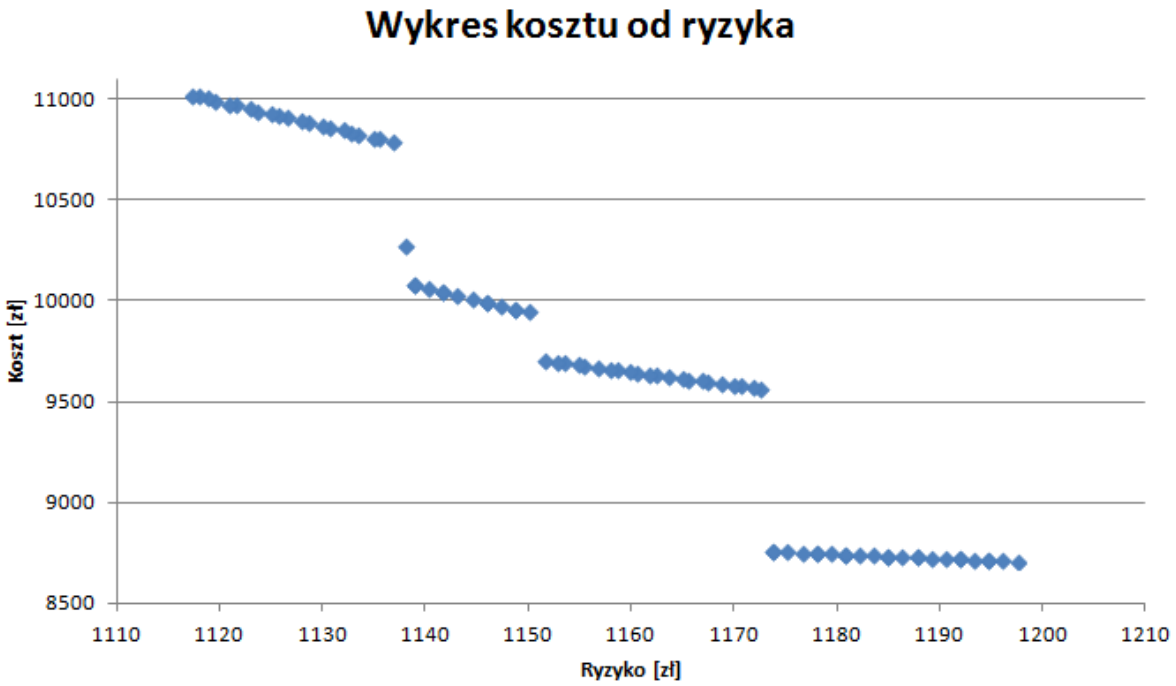
Gdzie:
- n - liczba generowanych scenariuszy,
- μ oraz Σ zmienne podane we wzorze (2.1),
- *df* - liczba stopni swobody równa 4,
- *lower* - wektor dolnej granicy przedziału równy [20, 20, 20],
- *upper* - wektor górnej granicę przedziału równy [50, 50, 50],
- *rejection* - domyślny algorytm.

3.1 Zbiór rozwiązań efektywnych

Liczba wygenerowanych scenariuszy wyniosła 100.

Zadanie zostało rozwiązane dwustopniowo, wpierw równanie ryzyka zostało ograniczone z góry przez liczbę, a koszt pracy maszyn był minimalizowany.
Następne minimalizowane było ryzyko, z kolei koszt pracy maszyn został ograniczony.

Wraz ze zmianą wartości liczb ograniczających wyrażenia, otrzymany został wykres przedstawiony na rysunku 3.1.



Rysunek 3.1: Wykres 90 rozwiązań efektywnych.

Powyższy wykres ma charakterystykę zbliżoną do wykresu hiperbolicznego, ponieważ zależność ryzyka od kosztu jest odwrotnie-proporcjonalna.

Na wykresie 3.1 widzimy iż rozwiązania efektywne są odcinkami liniowe, jest to spowodowane ograniczaniem zbioru przez nierówności liniowe.

Uskoki przedstawione na rysunku zostały spowodowane liczbą godzin pracy większej od 100, a co za tym idzie zmianą wartości binarnej na 1. Konsekwencją tego jest brak zniżki za każdą godzinę pracy, czyli większy koszt pracy maszyny.

	Punkt utopii	Punkt nadiru
Koszt	8708,97	11020,30
Ryzyko	1117,36	1197,51

Rysunek 3.2: Punkt utopii oraz nadiru.

3.2 Dominacja stochastyczna I rzędu

$$Y' \succeq_{IFSD} Y'' \quad \Longleftrightarrow \quad -Y' \succeq_{FSD} -Y'' \tag{3.4}$$

Na podstawie prawdziwości powyższego wyrażenia stworzony został wykres 3.3 przedstawiający dominacje stochastyczną pierwszego rzędu dla 3 rozwiązań efektywnych w oparciu o odwrotności dystrybuant.
Zadanie ma na celu minimalizacje funkcji kosztu dla 100 scenariuszy z jednakowym prawdopodobieństwem $\frac{1}{100}$ sprowadzając zmienne losowe do postaci loterii o jednakowo praw-

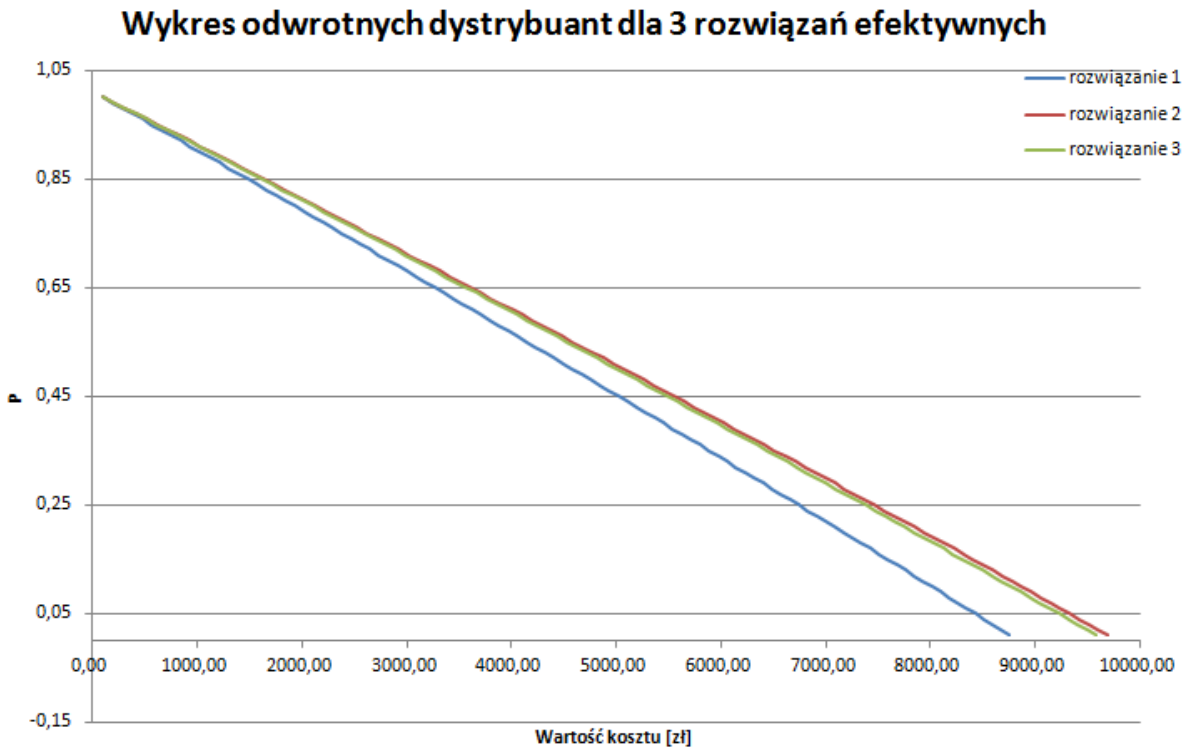
dopodobnych losach.

Zastosowane rozwiązania efektywne:

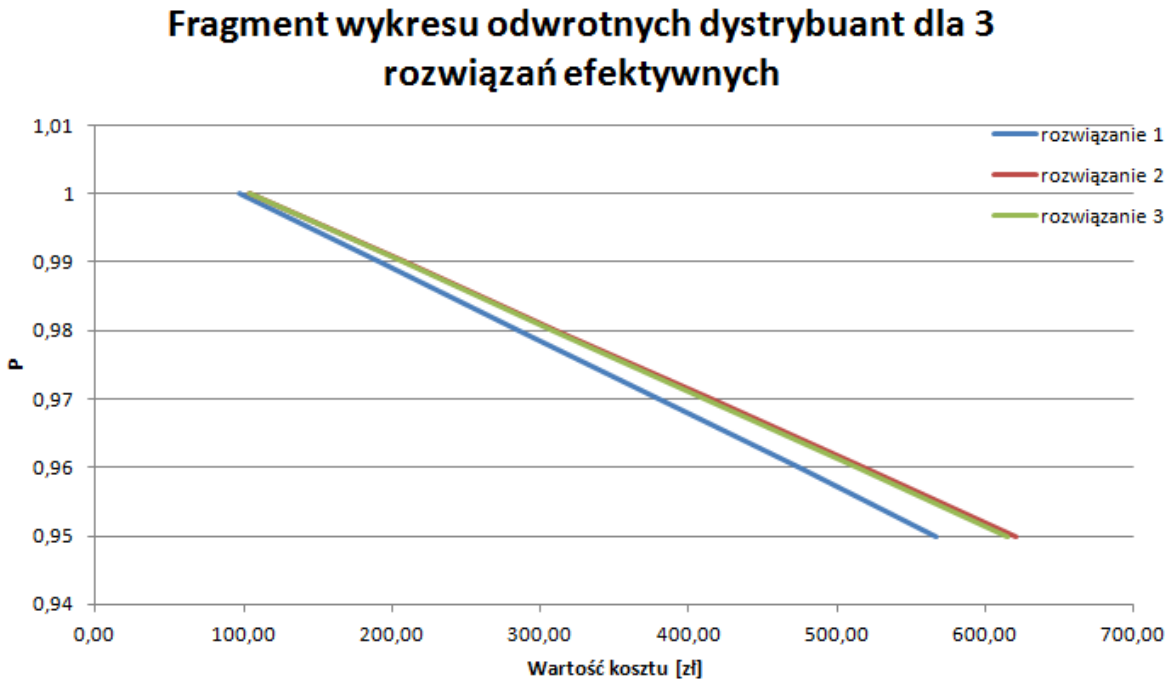
rozwiązanie 1
koszt = 8747,91
ryzyko = 1179,37

rozwiązanie 2
koszt = 9684,18
ryzyko = 1154,78

rozwiązanie 3
koszt = 9582,09
ryzyko = 1169,99



Rysunek 3.3: Wykres rozwiązań efektywnych.



Rysunek 3.4: Wykres rozwiązań efektywnych.

Na podstawie fragmentu wykresu przedstawionego powyżej widzimy, iż rozwiązania 2 oraz 3 przecinają się w lewym górnym obszarze - dla niskich wartości kosztu, a następnie wraz ze wzrostem kosztu rozchodzą się, co czyni je nieporównywalnymi.
Z kolei rozwiązanie 1 nie przecina się z pozostałymi rozwiązaniami oraz jego odwrotna dystrybuanta osiąga najniższe wartości, z czego wynika jego dominacja w sensie FSD rozwiązań 2 oraz 3.

4 Testy

4.1 Całkowita liczba sztuk

Po zastosowaniu ograniczeń zdefiniowanych nierównościami (2.3-2.5) każdorazowo otrzymywana została całkowita liczba wyprodukowanych sztuk.
Z kolei przy nie uwzględnieniu danych ograniczeń liczba sztuk dla tego samego modelu nie zawsze była liczbą całkowitą.

Przykład:

1. Nie uwzględniając ograniczeń:

Otrzymany koszt: 8652,80.

Liczba sztuk każdego wyprodukowanego podzespołu:

Maszyna	Liczba wyprodukowanych sztuk				
	A	B	C	D	E
M1	0,00	60,00	17.8(3)	0,00	0,00
M2	0,00	0,00	42.1(6)	35,00	0,00
M3	60,00	0,00	0,00	85,00	0,00

Rysunek 4.1: Liczba sztuk wyprodukowanego podzespołu podzespołu nieuwzględniając ograniczeń (2.3-2.5).

2. Uwzględniając ograniczenia:

Otrzymany koszt: 8653,48.

Liczba sztuk każdego wyprodukowanego podzespołu:

Maszyna	Liczba wyprodukowanych sztuk				
	A	B	C	D	E
M1	0,00	60,00	18,00	0,00	0,00
M2	0,00	0,00	42,00	35,00	0,00
M3	60,00	0,00	0,00	85,00	0,00

Rysunek 4.2: Liczba sztuk wyprodukowanego podzespołu podzespołu uwzględniając ograniczenia.

Na podstawie powyższych rysunków zostało przedstawione poprawne wymuszanie liczby całkowitych wyprodukowanych sztuk, mimo wzrostu kosztu.

4.2 Zmienna binarna

Zmienna $bool[k]$ wynosi 1, gdy w nierównościach (2.12-2.13) liczba godzin pracy k tej maszyny jest większa od 100.
Nierówność (4.1) przedstawia ograniczenia zmiennej $hours[k]$ dla tego przypadku.

$$\bigwedge_{bool[k]=1} \quad 100 \leq hours[k] \leq 100 + M \tag{4.1}$$

Z kolei $bool[k]$ przyjmuje wartość równą 0, dla liczby godzin pracy k-tej maszyny mniejszej bądź równej 100.
Ograniczenia przyjmują postać:

$$\bigwedge_{bool[k]=0} \quad 0 \leq hours[k] \leq 100 \tag{4.2}$$

Następnie zostało sprawdzone czy nierówności (2.15-2.18) odpowiednio zmieniają wartość zmiennej *costExtra* na 20% z wartości *cost* w przypadku zmiennej binarnej równej 1. Postać ograniczeń:

$$\bigwedge_{bool[k]=1} \quad L * bool[k] \leq costExtra[k] \leq U * bool[k] \tag{4.3}$$

$$\bigwedge_{bool[k]=1} \quad 0,2 * cost[k] * godz[k] \leq costExtra[k] \leq 0,2 * cost[k] * godz[k] \implies costExtra[k] = 0,2 * cost[k] * godz[k] \tag{4.4}$$

Dodatkowo sprawdzone zostały rezultaty dla zmiennej binarnej równej 0, czyli dla liczby godzin mniejszej bądź równej 100, która pociąga za sobą wartość *costExtra* równą 0, która jest pożądana w tym zadaniu ze względu na minimalizacje kosztów.

$$\bigwedge_{bool[k]=0} \quad 0,2 * cost[k] * godz[k] - U \leq costExtra[k] \leq 0,2 * cost[k] * godz[k] \tag{4.5}$$

$$\bigwedge_{bool[k]=0} \quad 0 \leq costExtra[k] \leq 0 \implies costExtra[k] = 0 \tag{4.6}$$

Testy zaimplementowane w środowisku *NetBeans* przebiegły poprawnie dla wymuszenia odpowiednio dla k-tej maszyny poniższego ograniczenia:

$$hours[k] \geq 101 \tag{4.7}$$

4.3 Generowane scenariusze

Zgodnie z mocnym prawem wielkich liczb, jeżeli $(\xi_{kn})_{n\in\mathbb{N}}$ jest ciągiem niezależnych zmiennych losowych o tym samym rozkładzie oraz $\mathbb{E}|R_k| < \infty$ dla k-tej maszyny.

$$\bigwedge_{k} \quad \frac{1}{n}\sum_{i=1}^{n} \xi_{ki} \xrightarrow{p.n.} \mathbb{E}(R_k) \tag{4.8}$$

Wzór (4.3) oznacza, że wraz ze wzrostem liczby wygenerowanych scenariuszy ich średnia zbiega do wartości oczekiwanej danej maszyny.

Test został przeprowadzony w środowisku *NetBeans*, gdzie za pomocą wartości bezwzględnej z różnicy wartości oczekiwanej oraz średniej wygenerowanych scenariuszy zaobserwowano zbieganie do zera wraz ze wzrostem liczby wygenerowanych scenariuszy.

Wyniki testu:

1. Dla liczby scenariuszy równej 20:
[0.75; 0,64; 0,63]

2. Dla liczby scenariuszy równej 100:
[0,05; 0,52; 0,01]

3. Dla liczby scenariuszy równej 10000:
[0,003; 0,001; 0,001]

5 Wnioski

Na podstawie dowiedzionej dominacji stochastycznej pierwszego rzędu wynika, iż wykorzystana średnia różnica Giniego jako miara ryzyka nie jest efektywna ze względu na dyspersje, tzn. rozproszenie spowodowaną różnicą realizacji scenariuszy generowanych losowo.

Na podstawie przeprowadzonych testów wynikło, iż implementacja modelu przebiegła zgodnie z oczekiwaniem.

Kod R

```
mu <- c(45, 35, 40)
sigma <- matrix(c(1, -2, -1, -2, 36, -8, -1, -8, 9), nrow=3, ncol=3)

data <- rtmvt(n=10000, mean=mu, sigma=sigma, df = 4, lower = c(20,20,20),
    upper=c(50,50,50))

data

write.table(data, "d:/R/data10000.txt", sep="\t", col.names = F, row.names = F)

Er1 <- c(45 +
    1*(gamma(3/2)*((4+(-25)^2)^(-3/2)-(4+5^2)^(-3/2))*4^2)/(2*(pt(5,4)-pt(-25,4))*gamma(2)*g
Er1

Er2 <- c(35 +
    2*(gamma(3/2)*((4+(-5/2)^2)^(-3/2)-(4+(15/2)^2)^(-3/2))*4^2)/(2*(pt(15/2,4)-pt(-5/2,4))*
Er2

Er3 <- c(40 +
    3*(gamma(3/2)*((4+(-20/3)^2)^(-3/2)-(4+(10/3)^2)^(-3/2))*4^2)/(2*(pt(10/3,4)-pt(-20/3,4)
Er3

write.table(Er1, "d:/R/er1.txt", sep="\t", col.names = F, row.names = F)
write.table(Er2, "d:/R/er2.txt", sep="\t", col.names = F, row.names = F)
write.table(Er3, "d:/R/er3.txt", sep="\t", col.names = F, row.names = F)
```

Kod Java

```
package wdwr;


import java.io.IOException;



/**
 *
 * @author Aisolug
 */
public class WDWR {
```

```
    /**
     * @param args the command line arguments
     * @throws java.io.IOException
     */
    public static void main(String[] args) throws IOException {
        model x = new model();

    }
}
```

```
package wdwr;

import ilog.cplex.*;

import ilog.concert.*;
import java.io.IOException;
import java.util.ArrayList;

/**
 *
 * @author Aisolug
 */
public class model {

    double[] cost = {44.9745448834606, 35.2347110553482, 39.8285917033513};
    double[][] efficiency = {
        {0.85, 1.3, 0.65, 1.5, 0.4},
        {0.65, 0.8, 0.55, 1.5, 0.7},
        {1.2, 0.95, 0.35, 1.7, 0.4}
    };
    int M = 100;
    double[] costAvg;

    public model() throws IOException {

        ReadFile inFile = new ReadFile();
        ArrayList<ArrayList<Double>> scenarios = inFile.generateScenarios();
        ArrayList<ArrayList<Double>> results = new ArrayList();

        costAvg = inFile.averageScenarios();
        /*
        Test scenariuszy
        for (int k = 0; k < 3; k++) {
```

```
     System.out.println(cost[k] - costAvg[k]);
}
 */
//for (int step = 0; step <80 ; step++){
try {

    IloCplex cplex = new IloCplex();
    /*

    i - numer podzespolu
    j - numer wymogu odnosnie ilosci sztuk danego podzespolu
    k - numer maszyny
    hours - suma godzin danej maszyny
    efficiency - wydajnosc
    cost - koszty danej maszyny
    M - duza liczba (180-100)

    inicjalizacja zmiennych
    3-elementowy wektor boolean dla kazdej z maszyn */
    IloNumVar[] bool = new IloNumVar[3];

    //macierz 3x5 sztuk
    IloNumVar[][] items = new IloNumVar[3][5];
    IloNumVar[] costExtra = cplex.numVarArray(3, 0.0,
        Double.MAX_VALUE);//jezeli to zakres przyjmowanych zmiennych to
        blad
    IloIntVar[][] intItems = new IloIntVar[3][5];

    for (int k = 0; k < 3; k++) {
        items[k] = cplex.numVarArray(5, 0.0, Double.MAX_VALUE);
        intItems[k] = cplex.intVarArray(5, 0, Integer.MAX_VALUE);
        bool[k] = cplex.boolVar();
    }

    /*
    nadanie ograniczen
    godz - suma godzin danej maszyny <= 180
    intemsEnough - ilosc kazdego z podzespolow A,B oraz C >= 60 oraz
        D+E >= 120
     */
    IloLinearNumExpr[] itemsEnough = new IloLinearNumExpr[4];
    IloLinearNumExpr[] allMachine = new IloLinearNumExpr[3];

    IloNumExpr model = cplex.numExpr();
```

```
IloNumExpr[] hours = new IloNumExpr[3];

for (int j = 0; j < 4; j++) {
    itemsEnough[j] = cplex.linearNumExpr();
}
for (int k = 0; k < 3; k++) {

    hours[k] = cplex.numExpr();

    allMachine[k] = cplex.linearNumExpr();

    for (int i = 0; i < 5; i++) {

        hours[k] = cplex.sum(hours[k], cplex.prod(1 /
            efficiency[k][i], items[k][i]));
        //wymuszanie zmiennej calkowitej liczby sztuk
        cplex.addLe(intItems[k][i], items[k][i]);
        cplex.addGe(intItems[k][i], cplex.sum(items[k][i], -1));
        cplex.addEq(intItems[k][i], items[k][i]);
        allMachine[k].addTerm(1, items[k][i]);
        if (i < 3) {
            //dla A,B oraz C
            itemsEnough[i].addTerm(1, items[k][i]);
        } else {
            //D+E
            itemsEnough[3].addTerm(1, items[k][i]);
        }
    }

    //suma godzin <= 180
    cplex.addLe(hours[k], 180);
    cplex.addGe(allMachine[k], 1);
    //czy suma godzin <= 100 jezeli tak to bool danej maszyny = 0

    cplex.addLe(hours[k], cplex.sum(100, cplex.prod(M, bool[k])));
    //godz<=100 + Mu
    //godz>=100u
    cplex.addGe(hours[k], cplex.prod(100, bool[k]));

    cplex.addGe(costExtra[k], cplex.prod(bool[k], 0.2 *
        costAvg[k]));
    cplex.addLe(costExtra[k], cplex.prod(bool[k], 36 * costAvg[k]));
    //z<=bool*50

    //z>=bool*5
```

```
        cplex.addLe(costExtra[k], cplex.prod(hours[k], 0.2 *
            costAvg[k]));
        //z<=0.2koszt + L(u-1)

        cplex.addGe(cplex.sum(costExtra[k], cplex.prod(36 * costAvg[k],
            cplex.sum(1, cplex.negative(bool[k])))),
            cplex.prod(hours[k], 0.2 * costAvg[k]));

        //cost -zad1, costAvg zad2
        model = cplex.sum(model, cplex.sum(cplex.prod(0.8 * costAvg[k],
            hours[k]), costExtra[k]));

    }

    //cd nadania ograniczen ilosci podzespolu A,B,C min 60 sztuk
    for (int j = 0; j < 3; j++) {
        cplex.addGe(itemsEnough[j], 60);

    }
    //cd nadania ograniczen ilosci podzespolu D+E min 120 sztuk
    cplex.addGe(itemsEnough[3], 120);

    //minimalizacja sumy kosztu dla kazdej z maszyn (liczba
        godzin*cena), jezeli bool danej maszyny =0 to koszty spadaja o
        20%
    IloNumExpr riskGini = cplex.numExpr();

    for (int l = 0; l < scenarios.size(); l++) {
        for (int n = 0; n < scenarios.size(); n++) {
            for (int k = 0; k < 3; k++) {

                riskGini = cplex.sum(riskGini,
                    cplex.prod(cplex.abs(cplex.sum(cplex.sum(cplex.prod(hours[k],
                    0.8 * scenarios.get(l).get(k)), costExtra[k]),
                        cplex.negative(cplex.sum(cplex.prod(hours[k],
                            scenarios.get(n).get(k)), costExtra[k])))),
                            1.0 / (2 * scenarios.size() *
                            scenarios.size())));
            }
        }

    }
    /* Test zmiennej binarnej
     cplex.addGe(hours[0],101);
     cplex.addGe(hours[1],101);
```

```
     cplex.addGe(hours[2],101);
     */
    cplex.addLe(riskGini, 1155);
    cplex.addMinimize(model);

    //cplex.addMinimize(cplex.prod(1/400, riskGini));
    //wyswietl koszt oraz sztuki
    if (cplex.solve()) {
        results.add(new ArrayList<>());
        results.get(results.size() - 1).add(cplex.getValue(model));
        results.get(results.size() - 1).add(cplex.getValue(riskGini));
        System.out.println(cplex.getValue(model));
        System.out.println(cplex.getValue(riskGini));

        for (int k = 0; k < 3; k++) {
            System.out.println(cplex.getValue(hours[k]));

            for (int i = 0; i < 5; i++) {
                System.out.println("sztuki " +
                    cplex.getValue(items[k][i]));
            }
            System.out.println(cplex.getValue(costExtra[k]));
            System.out.println(cplex.getValue(bool[k]));
        }
        double[] costScenario = new double[scenarios.size()];
        for (int l = 0; l < scenarios.size(); l++) {
            double suma = 0.0;
            for (int k = 0; k < 3; k++) {
                suma += cplex.getValue(hours[k]) *
                    scenarios.get(l).get(k) * (0.8 + 0.2 *
                    cplex.getValue(bool[k]));
            }
            costScenario[l] = suma / scenarios.size();
        }
        dystrybuanty fsd = new dystrybuanty();
        fsd.dystryb(costScenario);
    } else {
        System.out.println("Model not solved");
    }

} catch (IloException exc) {
    System.out.print(exc);
}
//}
inFile.saveFile(results);
```

```
    }

}
```

```
package wdwr;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Locale;
import java.util.Scanner;


/**
 *
 * @author Aisolug
 */
final class ReadFile {

    public ReadFile() throws FileNotFoundException, IOException {
    }
    ArrayList<ArrayList<Double>> scenarios;

    //wczytanie scenariuszy
    public ArrayList<ArrayList<Double>> generateScenarios() throws
        FileNotFoundException {

        try (Scanner fileScanner = new Scanner(new
            File("D:\\R\\data100.txt")).useLocale(Locale.US)) {
            scenarios = new ArrayList<>();
            while (fileScanner.hasNextLine()) {
                scenarios.add(new ArrayList<>());
                for (int k = 0; k < 3; k++) {

                    scenarios.get(scenarios.size() -
                        1).add(fileScanner.nextDouble());
                }

            }
            fileScanner.close();
        }
```

```
        return scenarios;
    }
//liczenie sredniej ze scenariuszy
    public double[] averageScenarios() {
        double[] costAvg = new double[3];
        for (int k = 0; k < 3; k++) {
            for (int l = 0; l < scenarios.size(); l++) {
                costAvg[k] += scenarios.get(l).get(k);
            }
            costAvg[k] /= scenarios.size();
        }

        return costAvg;
    }
//zapis danych do pliku
    public void saveFile(ArrayList<ArrayList<Double>> lista) throws
        IOException {
        FileWriter fileWriter;
        fileWriter = new FileWriter("D:\\R\\result2.csv");
        //zapis r "cost"
        fileWriter.append("cost");
        fileWriter.append(',');
        //zapisa r max
        fileWriter.append(String.valueOf("risk"));
        //oddzielenie od reszty
        fileWriter.append('\n');
        fileWriter.append('\n');
        for (int j = 0; j < lista.size(); j++) {

            for (int i = 0; i < 2; i++) {
                //zapisywanie kolejnych wartosci tak, ze odpowiednie wartosci
                    zmiennych w odpowiednich kolumnach
                fileWriter.append(String.valueOf(lista.get(j).get(i)));
                fileWriter.append(",");
            }
            fileWriter.append('\n');
        }
        fileWriter.flush();
        fileWriter.close();

    }
}
```